OSCER Note

Virtual Environment Management
python -m venv myenv: Creates a virtual environment named myenv.

source ~/myenv/bin/activate: Activates the myenv virtual environment.

Mamba Package Management (Mamba is a faster, more efficient alternative to Conda)
module load Mamba: Loads the Mamba environment management tool.

mamba init: Initializes Mamba for shell integration.

mamba create --name myenv: Creates a new conda environment named myenv.

mamba activate: Activates a previously created conda environment.

mamba env list: Lists all available environments.

mamba clean --all: Cleans all temporary files and caches in Mamba.

mamba remove --name myenv --all: Removes the myenv environment and all its contents.

Job and Quota Management
getquota: Displays the storage quota and usage.

squeue -u username: Lists all jobs of a specific user in the queue.

sinfo -p partition_name: Shows the status of a specific partition.

sbatch script_name.sbatch: Submits a job script to the scheduler.

tail -f <batch_file_output>: Continuously monitors the output of a batch job.

scontrol <jobid>: Manages or inspects details of a specific job.

*This note was made by Nam Huynh

scancel <jobid>: Cancels a specific job by job ID.

Module Management
module purge: Unloads all loaded modules.

module load <application name>: Loads a specific module or application.

module unload <application name>: Unloads a specific module or application.

module list: Lists all currently loaded modules.

module avail: Lists all available modules.

File and Cache Cleanup
rm -rf ~/.cache/*: Deletes all cached files.

rm -rf ~/.conda: Removes the .conda directory and its contents.

rm -rf ~/.local/lib/python*: Deletes local Python library files.

CUDA and cuDNN

To view all loadable CUDA modules in OSCER, type:
module avail cuda/

or for cuDNN:

module avail cudnn/

*This note was made by Nam Huynh

Example of batch scripts:

1/ To use GPU

```
#!/bin/bash
#
#SBATCH --partition=gpu_a100
#SBATCH --output=test_run_%J_stdout.txt
#SBATCH --error=test_run_%J_stderr.txt
#SBATCH --ntasks=1
#SBATCH --mem=60G
#SBATCH --time=12:00:00
#SBATCH --gres=gpu:2
#SBATCH --exclusive

# Initialize Conda
module load Mamba
mamba init
module load CUDA


# Your code here

```

2/ To use OSCER on VScode

```
#!/bin/bash
#
#SBATCH --partition=vscode
#SBATCH --output=vscode_%J_stdout.txt
#SBATCH --error=vscode_%J_stderr.txt
#SBATCH --ntasks=1
#SBATCH --mem=20G
#SBATCH --time=12:00:00


# Initialize Conda
module load Mamba
mamba init
module load CUDA


# Your code here

```

Scripts explanation:

#SBATCH --partition=gpu_a100: Specifies the partition (queue) where the job will be submitted. In this case, it's the gpu_a100 partition, meaning the job will use nodes with A100 GPUs.

#SBATCH --output=test_run_%J_stdout.txt: Sets the file where standard output (stdout) will be written. %J is a placeholder that will be replaced by the job ID. The output will be saved in a file named test_run_<job_id>_stdout.txt.

#SBATCH --error=test_run_%J_stderr.txt: Sets the file where standard error (stderr) messages will be written. %J will be replaced with the job ID, and the error output will be saved in test_run_<job_id>_stderr.txt.

#SBATCH --ntasks=1: Specifies the number of tasks to be launched for the job, in this case, 1 task (a single job instance) or just consider it as not running parralel jobs.

#SBATCH --mem=60G: Requests 60 GB of memory for the job.

*This note was made by Nam Huynh

#SBATCH --time=12:00:00: Specifies the maximum run time for the job as 12 hours. After this time, the job will be terminated if it hasn't finished.

#SBATCH --gres=gpu:2: Requests 2 GPUs for the job using Slurm's generic resource (gres) system.

#SBATCH --exclusive: This requests exclusive access to the entire node, meaning no other jobs will share the same node resources while this job is running.

*If you want to run a batch file A on the same node with batch file B

_Check for batch file B's node using: squeue -u username

_Put this on batch file A: #SBATCH --nodelist=<node>

*This note was made by Nam Huynh

DISC

1/ Apply for DISC GPU partition access to use these 2 GPUs: disc, disc_dual_a100 and use OURdisk which is like a regular external hard drive with no “expiration date”.

The path to your directory on DISC's OURdisk is:

/ourdisk/hpc/disc/USERNAME/auto_archive_notyet/tape_2copies/

If you do not wish your data to be automatically archived, you can put your files in the following directory:

/ourdisk/hpc/disc/nam/dont_archive/

Files and directories created in the dont_archive directory will NOT be automatically archived.

_https://www.ou.edu/disc/resources

_https://www.ou.edu/disc/about/people/disc-membership

_Apply for access to the DISC supercomputer resources: https://ousurvey.qualtrics.com/jfe/form/SV_ac6ajVyfgZXeWy2

Additional things to consider:

_For more storage in home directory: Email support@oscer.ou.edu for 20GB more.

_Submitting batch job on schooner login nodes (i.e. Via mobaxterm, putty, or cmd).

*This note was made by Nam Huynh

Using Ollama (A platform designed to run large language models (LLMs)):

A. Starting Ollama "serve" on OSCER to run Ollama commands

1/ rm ollama (executable downloaded file) # if ollama already exists

2/ download ollama on OSCER: wget https://github.com/ollama/ollama/releases/download/v0.3.6/ollama-linux-amd64

3/ change ollama file name and set it to be executable:

mv ollama-linux-amd64 ollama

chmod 755 ollama

4/ start Ollama server:

sbatch startOllamaServer.batch (need a GPU) to run this command: ./ollama serve

B. Upload the model to Ollama

1/ Sign-up and create a model on Ollama:

https://www.ollama.ai/signup

https://www.ollama.ai/signup

2/ create model locally before pushing to Ollama:

sbatch uploadToOllama.batch (need a GPU) (remember to change nodelist to be the same with startOllamaServer.batch) to run this command: ollama create <ollama-username>/<model-name> -f /path/to/Modelfile

3/ get the public key ssh from OSCER to put it on Ollama key:

cat ~/.ollama/id_ed25519.pub

https://ollama.com/settings/keys

4/ push the model to Ollama:

sbatch ollamapush.batch (need a GPU) (remember to change nodelist to be the same with startOllamaServer.batch) to run this command: ollama push <ollama-username>/<model-name>

C. Deploy the model (on local machine)

1/ Download Ollama to local-machine from this: https://github.com/ollama/ollama

2/ Deploy desired model using this command on the terminal: ollama run <model_name>

Batch files for Ollama

startOllamaServer.batch

```
#!/bin/bash
#
#SBATCH --partition=disc
#SBATCH --output=startServer_run_%J_standout.txt
#SBATCH --error=startServer_run_%J_stderr.txt
#SBATCH --ntasks=1
#SBATCH --mem=60G
#SBATCH --time=6:00:00
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=10

module --ignore-cache load "cuDNN/8.7.0.84-CUDA-12.1.0"
module load GCC/11.3.0

export HF_TOKEN=##############################

./ollama serve


```

*This note was made by Nam Huynh

uploadToOllama.batch

```
#!/bin/bash
#
#SBATCH --partition=disc
#SBATCH --output=uploadToOllama_run_%J_standout.txt
#SBATCH --error=uploadToOllama_run_%J_stderr.txt
#SBATCH --ntasks=1
#SBATCH --mem=60G
#SBATCH --time=6:00:00
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=10
#SBATCH --nodelist=c856
##SBATCH --exclusive

#module load CUDA
#export HF_HOME=/ourdisk/hpc/disc/nam/auto_archive_notyet/tape_2copies/hf_cache
#module load Python/3.10.4-GCCcore-11.3.0

#source ../../myenv/bin/activate
module --ignore-cache load "cuDNN/8.7.0.84-CUDA-12.1.0"
module load GCC/11.3.0

export HF_TOKEN=##############################

#./ollama serve
#sleep 10
./ollama create NamHuynh1308/llama3.1-python-sorting -f Modefile


```

ollamapush.batch

```
#!/bin/bash
#
#SBATCH --partition=disc
#SBATCH --output=pushOllama_run_%J_standout.txt
#SBATCH --error=pushOllama_run_%J_stderr.txt
#SBATCH --ntasks=1
#SBATCH --mem=60G
#SBATCH --time=6:00:00
#SBATCH --gres=gpu:1
#SBATCH --cpus-per-task=10
#SBATCH --nodelist=c856
##SBATCH --exclusive

#module load CUDA
#export HF_HOME=/ourdisk/hpc/disc/nam/auto_archive_notyet/tape_2copies/hf_cache
#module load Python/3.10.4-GCCcore-11.3.0

#source ../../myenv/bin/activate
module --ignore-cache load "cuDNN/8.7.0.84-CUDA-12.1.0"
module load GCC/11.3.0

export HF_TOKEN=##############################

./ollama push NamHuynh1308/llama3.1-python-sorting


```

*This note was made by Nam Huynh